

## **BAB III**

### **LANDASAN TEORI**

#### **3.1 Pengertian Sistem Rekomendasi**

Sistem rekomendasi adalah suatu sistem yang merekomendasikan sebuah informasi kepada pengguna atau menduga apa yang akan dilakukan pengguna untuk mencapai tujuannya (Kurniawan, 2016). Sistem rekomendasi akan membimbing pengguna untuk menemukan produk yang relevan dan berguna dari banyaknya produk yang tersedia (Prasetya, 2017). Hampir semua area bisnis sudah menerapkan konsep sistem rekomendasi, yaitu dimana seorang konsumen memerlukan informasi untuk membuat suatu keputusan (Utomo & Anggriawan, 2015).

#### **3.2 Pengertian Aplikasi Sistem Informasi**

Sistem informasi adalah suatu sistem dalam suatu organisasi yang mempertemukan kebutuhan-kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi serta menyediakan pihak luar tertentu dengan laporan-laporan yang dibutuhkan (Kurniawan & Irani, 2015).

#### **3.3 Pengertian Aplikasi *Desktop***

Aplikasi desktop adalah aplikasi yang berjalan lokal di dalam lingkungan *desktop* dan hanya dapat diakses oleh pengguna *desktop* yang mengeksekusinya (Adiputra & Mustofa, 2015).

### 3.4 Pengertian Algoritma Apriori

Apriori adalah suatu algoritma untuk melakukan pencarian *frequent itemset* dengan menggunakan teknik *association rule*. Algoritma Apriori biasanya digunakan pada penerapan *market basket analysis* untuk mencari aturan asosiasi yang memenuhi batas *support* dan *confidence*. Pada pencarian aturan asosiasi dibutuhkan parameter supaya aturan yang didapat akurat. Parameter yang digunakan untuk membentuk suatu *rules* adalah:

a. *Support*

*Support* adalah suatu ukuran yang menunjukkan seberapa besar tingkat dominasi barang/*itemset* dari keseluruhan transaksi yang ada. Untuk menghitung nilai *support* menggunakan rumus :

$$Support(A) = \frac{\text{jumlah transaksi mengandung A}}{\text{total transaksi}} \quad (3,1)$$

Untuk menghitung prosentase *support* dari suatu item menggunakan rumus :

$$Support(A) = \frac{\text{jumlah transaksi mengandung A}}{\text{total transaksi}} \times 100\% \quad (3,2)$$

b. *Confidence*

*Confidence* adalah suatu ukuran yang menunjukkan hubungan kondisional antara dua barang (contoh misalkan seberapa sering paku A dibeli jika orang membeli paku B). Untuk menghitung nilai *confidence*, didapat dengan menggunakan rumus :

$$Confidence(A \rightarrow B) = \frac{\text{jumlah transaksi mengandung A dan B}}{\text{jumlah transaksi mengandung A}} \quad (3,3)$$

Nilai *confidence* dapat juga didapat dengan menggunakan rumus :

$$Confidence(A \rightarrow B) = \frac{Support(A,B)}{Support(A)} \quad (3,4)$$

Untuk menghitung nilai prosentase *confidence* dapat menggunakan rumus :

$$Confidence(A \rightarrow B) = \frac{Support(A,B)}{Support(A)} \times 100\% \quad (3,5)$$

Mengembangkan *frequent itemset* dengan dua *item*, dapat dengan menggunakan satu *item*, karena bila set satu *item* tidak dapat mencapai *minimum support*, maka setiap *itemset* yang dengan ukuran lebih besar juga tidak akan melebihi *minimum support*.

Algoritma Apriori memiliki dua proses utama, yaitu :

a. *Join* (penggabungan)

Proses penggabungan ini merupakan proses dimana setiap *item* dikombinasikan dengan *item* lain sampai tidak dapat terbentuk kombinasi lagi.

b. *Pruning* (pemangkasan)

Proses pemangkasan ini merupakan proses dimana hasil kombinasi *item* yang tadi telah digabungkan akan dipangkas berdasarkan *minimum support* yang telah ditentukan (Rabbany & Aripin, 2016).

Contoh apriori :

Toko Besi memiliki data transaksi sebagai berikut :

Tabel 3.1 Data Transaksi

<b>Id Transaksi</b>	<b>Barang</b>
1	Kuas, Semen, Pasir
2	Semen, Pasir, Batu Bata, Besi
3	Split, Batu Bata, Besi
4	Semen, Pasir

5	Semen, Pasir, Cat Tembok, Batu Bata, Besi
---	---

Misal nilai *minimum support* adalah 40%

#### Iterasi 1

Untuk 1-*itemset* hitung dan *scan database* untuk mendapatkan pola *frequent* dari *support*

Tabel 3.2 1-*itemset* (C1)

<i>Itemset</i>	<i>Support Count</i>	<i>Support</i>
Semen	4	80%
Pasir	4	80%
Batu Bata	3	60%
Besi	3	60%
Split	1	20%
Cat Tembok	1	20%
Kuas	1	20%

dapatkan k-*itemset* dari *support* yang memenuhi *minimum support*, kemudian pilih k-*itemset* sebagai pola *frequent* tinggi.

Tabel 3.3 Pola *Frequent* (F1)

<i>Itemset</i>	<i>Support Count</i>	<i>Support</i>
Semen	4	80%
Pasir	4	80%
Batu Bata	3	60%
Besi	3	60%

#### Iterasi 2

Pada iterasi sebelumnya pola *frequent* dari *support* telah didapatkan dari 1-*itemset*, untuk 2-*itemset*, generate k-*itemset* dari k-*itemset* iterasi sebelumnya dengan melakukan kombinasi dari k-*itemset* tersebut.

Tabel 3.4 Kombinasi dari *k-itemset* (C2)

<i>Itemset</i>
Semen, Pasir
Semen, Batu Bata
Semen, Besi
Pasir, Batu Bata
Pasir, Besi
Batu Bata, Besi

C2 adalah *itemset* dari kombinasi *k-itemset* dari iterasi sebelumnya, setelah didapatkan *k-itemset* tersebut, hitung masing-masing *item frequent* dan *scan database* dan dapatkan *frequent item* dari *support*.

Tabel 3.5 2-*itemset* (C2)

<i>Itemset</i>	<i>Support Count</i>	<i>Support</i>
Semen, Pasir	4	80%
Semen, Batu Bata	2	40%
Semen, Besi	2	40%
Pasir, Batu Bata	2	40%
Pasir, Besi	2	40%
Batu Bata, Besi	3	60%

Pengembangan algoritma apriori dengan memangkas *k-itemset* dengan menghitung *support* dari *itemset*, salah satu *itemset* yang tidak muncul dalam *database* {Batu Bata, Split} dari C2, sehingga dipangkas menjadi lebih menghemat *memory*.

Berikut tabel pola *frequent* tinggi diatas *minimum support* untuk 2-*itemset*

Tabel 3.6 Pola *frequent* tinggi (F2)

<i>Itemset</i>	<i>Support Count</i>	<i>Support</i>
Semen, Pasir	4	80%

Semen, Batu Bata	2	40%
Semen, Besi	2	40%
Pasir, Batu Bata	2	40%
Pasir, Besi	2	40%
Batu Bata, Besi	3	60%

Iterasi 3

Tabel 3.7 Kombinasi dari *k-itemset* (C3)

<b><i>Itemset</i></b>
Semen, Pasir, Batu Bata
Semen, Pasir, Besi
Pasir, Batu Bata, Besi

Tabel 3.8 3-*itemset* (C3)

<b><i>Itemset</i></b>	<b><i>Support Count</i></b>	<b><i>Support</i></b>
Semen, Pasir, Batu Bata	2	40%
Semen, Pasir, Besi	2	40%
Pasir, Batu Bata, Besi	40%	2

Kandidat 3-*itemset* yang telah memenuhi *minimum support*, *itemset* tersebut akan menjadi acuan untuk *k-itemset* selanjutnya

Tabel 3.9 3-*itemset* untuk pola *frequent* tinggi (F3)

<b><i>Itemset</i></b>	<b><i>Support Count</i></b>	<b><i>Support</i></b>
Semen, Pasir, Batu Bata	40%	2
Semen, Pasir, Besi	40%	2
Pasir, Batu Bata, Besi	40%	2

Iterasi 4

Tabel 3.10 Kombinasi 3-*itemset* untuk k-*itemset* (C4)

<i>Itemset</i>
Semen, Pasir, Batu Bata, Besi

Scan database untuk mendapatkan *itemset* dari *support*, *itemset* yang memenuhi *minimum support* dipilih sebagai pola *frequent* tinggi

Tabel 3.11 Pola *frequent* tinggi (F4)

Itemset	Support Count	Support
Semen, Pasir, Batu Bata, Besi	2	40%

Tidak ada lagi kombinasi yang bisa dibentuk untuk k-*itemset* berikutnya, proses berhenti, pola *frequent* tinggi yang ditemukan adalah “Semen, Pasir, Batu Bata, Besi”.

Kemudian membentuk *association rules* yang memenuhi syarat minimum dengan menghitung *confidence association rules*  $A \rightarrow B$

Pembentukan Aturan Asosiatif :

Tabel 3.12 Aturan Asosiatif

Jika membeli (A)	Maka membeli (B)	Support (AuB)	Support (A)	Confidence
Semen, Pasir, Batu Bata	Besi	40%	60%	66%
Besi, Pasir, Batu Bata	Semen	40%	80%	50%
Semen, Besi, Batu Bata	Pasir	40%	80%	50%

Besi, Pasir, Semen	Batu Bata	40%	60%	66%
Besi, Pasir	Batu Bata, Semen	40%	40%	100%
Besi, Batu Bata	Pasir, Semen	60%	80%	75%
Besi, Semen	Pasir, Batu Bata	40%	40%	100%
Pasir, Batu Bata	Besi, Semen	40%	40%	100%
Pasir, Semen	Besi, Batu Bata	80%	60%	75%
Batu Bata	Pasir, Besi	40%	40%	100%
Batu Bata	Pasir, Semen	40%	80%	50%
Besi	Pasir, Semen	60%	80%	75%
Besi	Pasir, Batu Bata	60%	40%	66%
Pasir	Besi, Batu Bata	80%	60%	75%
Pasir	Besi, Semen	80%	40%	50%
Pasir	Besi	80%	60%	75%
Pasir	Semen	80%	80%	100%
Pasir	Batu Bata	80%	60%	75%
Besi	Semen	60%	80%	75%
Besi	Batu Bata	60%	60%	100%
Batu Bata	Semen	60%	80%	75%



Dari aturan asosiatif yang didapat, bahwa dapat disimpulkan promo yang bisa direkomendasikan terbesar kepada pemilik toko adalah saat membeli besi, pasir, batu bata, dan semen maka mendapatkan diskon pada batu bata dan semen. Disini diambil karena memenuhi nilai *confidence* tertinggi, yaitu 100%. Kombinasi lain yang mendapat nilai *confidence* 100% adalah jika membeli besi dan semen maka membeli pasir dan batu bata, jika membeli pasir dan batu bata maka membeli besi dan semen, jika membeli batu bata maka membeli pasir dan besi, jika membeli pasir maka membeli semen, dan jika membeli besi maka membeli batu bata.

### 3.5 Pengertian C#

C# adalah bahasa pemrograman yang dibuat oleh Microsoft dan banyak digunakan oleh *developer* .NET untuk mengembangkan aplikasi dengan platform .NET. Bahasa C# adalah bahasa yang berorientasi *Object Oriented Programming* (OOP), memiliki banyak inti, mempunyai banyak kemiripan dengan bahasa C++, Java, dan VB. C# mengkombinasikan kekuatan dan efisiensi dari C++, sederhana dan kemudahan desain dari Java, dan bahasa yang sangat simpel seperti VB.

Bahasa C# biasanya digunakan untuk membangun berbagai macam jenis aplikasi, seperti aplikasi berbasis *desktop*, aplikasi berbasis *web*, dan aplikasi berbasis *web service*. C# menghilangkan beberapa hal yang bersifat kompleks yang sebelumnya terdapat pada bahasa pemrograman seperti C++ dan Java diantaranya yaitu *macro*, *templates*, *multiple inheritance* dan *virtual base classes*.

Bahasa pemrograman yang bersifat *Object Oriented* memiliki ciri yaitu *encapsulation*, *inheritance*, serta *polymorphism*. *Encapsulation* artinya dimana semua fungsi ditempatkan pada satu paket/*single package*. *Inheritance* artinya suatu cara terstruktur dari kode-kode pemrograman dan fungsi untuk menjadikan sebuah program baru dan berbentuk suatu paket. Kemudian

*polymorphism* artinya kemampuan untuk mengadaptasi apa yang diperlukan untuk dikerjakan. Ketiga sifat tersebut telah dimiliki oleh bahasa pemrograman C#.

Microsoft mengeluarkan produk-produk utamanya dengan dukungan *Framework .NET*, maka masa depan dari C# adalah sebagai salah satu bahasa pemrograman yang ada di dalam lingkungan *Framework .NET* akan lebih baik (Yulianto, 2013).

### 3.6 Pengertian Database

*Database* adalah sekumpulan tabel-tabel yang berisi data dimana merupakan kumpulan dari *field* atau kolom. *Record* dan *field* merupakan struktur file yang menyusun sebuah *database*. Data merupakan satuan informasi yang akan diolah, data akan dikumpulkan di dalam suatu *file database* sebelum nantinya diolah. Record adalah sebuah data yang isinya merupakan satu kesatuan, seperti nama *user* dan *password*. Setiap keterangan yang mencakup nama *user* dan *password* dinamakan satu *record*. Setiap *record* diberi nomor urut dan disebut nomor *record*. Filed adalah sub bagian dari *record* (Lasmana & Purba, 2013).

### 3.7 Pengertian MYSQL

MYSQL adalah suatu perangkat lunak *database* relasi (*Relational Database Management System*) seperti Oracle, Postgresql, MS SQL, dan sebagainya. Berdasarkan riset dinyatakan bahwa di platform web, baik untuk kategori *open source* dan umum, MYSQL adalah *database* yang paling banyak dipakai. Menurut pengembangnya, MYSQL telah dipasang di sekitar 3 juta komputer. MYSQL menggunakan SQL sebagai bahasa dasar untuk mengakses *databasenya*. MYSQL bersifat *free/gratis* dengan lisensi GNU *General Public License* (GPL). MYSQL termasuk dalam jenis RDMS (*Relational Database Management System*), maka dari itu pada MYSQL menggunakan istilah table, baris, dan kolom. Sebuah *database* di MYSQL sendiri mengandung satu atau sejumlah tabel (Prasetyo, et al., 2015).